

INFORMATICA

Hoy veremos...

- Sentencias condicionales:
 - Estructura `if ... else if ... endif`
 - Estructura `select case ... case ... default`
- Bucles iterativos:
 - Estructura `do ... enddo`

Antes:

- ¿Dudas práctica jueves pasado?

...

- Comentarios:

- División entera $(4/3) = 1$
- Exponentes $r^{**3.d0}$
- Inicializar según el tipo:

`real*8,parameter :: pi=3.14159265358979d0`

`real*8,parameter :: pi=acos(-1.d0)`

- Operador asignación ($a=b$)
- Intercambiar variables

Antes:

3.-Crea un nuevo proyecto (mismo procedimiento que antes) y usa el fichero principal2.f95 como programa principal. Corrige todos los errores y las advertencias que aparezcan cuando trates de compilarlo. Busca errores de programación no sintácticos.

```
program buscar errores

integer :: i,j,i_i

i=0.d0
j=1

real,parameter ::pi=3.1415926535
real*8  :: x,y,X,z

y = 3
z =1.7d0

X = y*i

k = y*z

write(*,*) 'el valor de X es ',x
write(*,*) 'Pi con 10 cifras decimales vale: ',pi

end program
```

Antes:

```
r\EIAE\Informatica 15-16\1ºCuatrimestre\Codigos\Clase_3 - [primer_proyecto]
er Documento Proyecto Construir Herramientas Ayuda
Guardar Guardar todo Revertir Cerrar Atrás Adelante Compilar Cons
estructuras.f90 Geometria_Computacional.f90 Aztec_file.f90 boundary_lay
1 program inicializar
2
3 real*8, parameter :: pi = 3.14159265358979d0
4 real*8, parameter :: pi_mal = 3.14159265358979
5 real*8, parameter :: pi2 = acos(-1.d0)
6 real*8, parameter :: pi2_mal = acos(-1.)
7
8 write(*,*) pi
9 write(*,*) pi_mal
10 write(*,*) pi2
11 write(*,*) pi2_mal
12
13 !write(*,*) (-2)**3d0 !Error
14 write(*,*) (-2)**3
15
16 end program
17
```

```
C:\Windows\system32\cmd.exe
3.1415926535897900
3.1415927410125732
3.1415926535897931
3.1415927410125732
-8
Presione una tecla para continuar . . . _
```

Recordamos estructura básica de un programa

```
program pepe  
  
    implicit none  
  
    ! Declaración de variables  
  
    ! Inicialización de variables  
  
    ! Cuerpo del programa  
  
end program pepe
```

Operadores Numéricos

- Aritméticos

| | |
|--------------|----|
| Suma | + |
| Resta | - |
| Producto | * |
| Cociente | / |
| Potenciación | ** |

- Relacionales

| | |
|-------------------|----|
| mayor que | > |
| menor que | < |
| mayor o igual que | >= |
| menor o igual que | <= |
| igual que | == |
| distinto que | /= |

Se usan en expresiones lógicas cuya resultado es VERDAD o MENTIRA

Operadores Numéricos

- Relacionales

```
program logicos
  real    :: a
  logical :: flag

  a = 0.d0

  write(*,*) (a==a)
  write(*,*) (a/=a)
  write(*,*) ((a/=a).or.(a==a))
  write(*,*) ((a/=a).and.(a==a))

  flag = ((a/=a).and.(a==a)).or.(a<a)

  write(*,*) flag

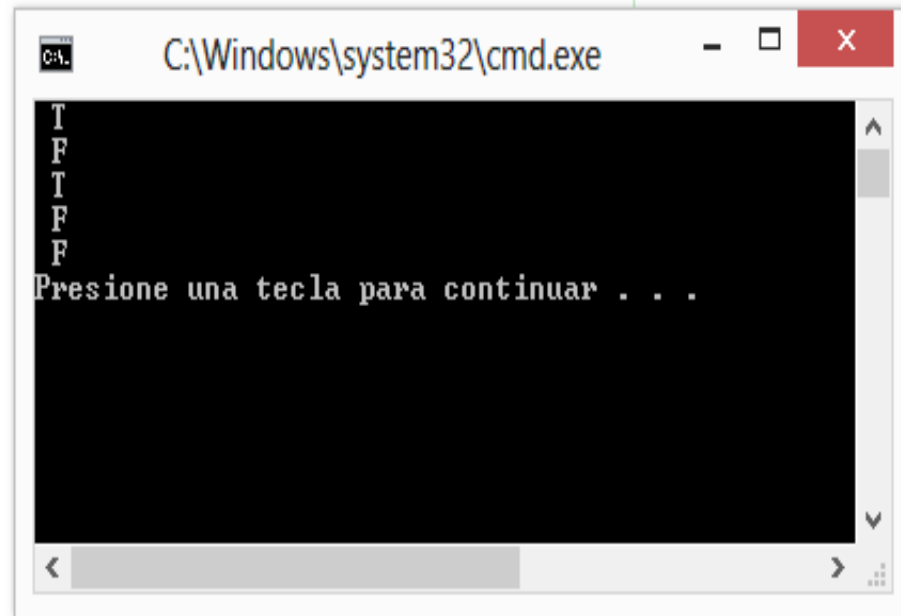
end program
```

El resultado de una operación lógica entre datos numéricos es una variable lógica (.TRUE. o .FALSE.)

Operadores Numéricos

- Relacionales

```
program logicos  
  
  real    :: a  
  logical :: flag  
  
  a = 0.d0  
  
  write(*,*) (a==a)  
  write(*,*) (a/=a)  
  write(*,*) ((a/=a).or.(a==a))  
  write(*,*) ((a/=a).and.(a==a))  
  
  flag = ((a/=a).and.(a==a)).or.(a<a)  
  
  write(*,*) flag  
  
end program
```



```
C:\Windows\system32\cmd.exe  
T  
F  
T  
F  
F  
Presione una tecla para continuar . . .
```

Operadores Numéricos

- Relacionales

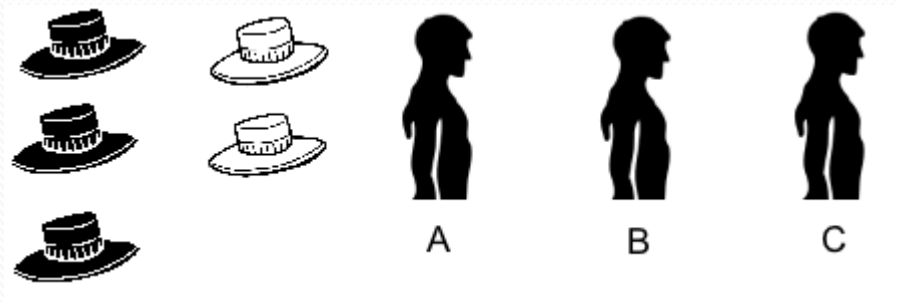
| | | |
|--------------------------------------|----------------------|---|
| $A == A$ | <code>.TRUE.</code> | T |
| $A \neq A$ | <code>.FALSE.</code> | F |
| $(A == A) \text{ .AND. } (A \neq A)$ | <code>.FALSE.</code> | F |
| $(A == A) \text{ .OR. } (A \neq A)$ | <code>.TRUE.</code> | T |

- Juegos lógicos

En una mesa hay tres sombreros negros y dos blancos.

Tres señores en fila india se ponen un sombrero al azar cada uno sin mirar el color.

Al cabo de un rato, el primero de la fila (C) que no ve ningún sombrero responde acertadamente de que color es el sombrero que tenía puesto.



¿Cuál es este color y cual es la lógica que uso para saberlo?

Operadores Numéricos

- Juegos lógicos



En un monasterio los monjes sólo se reúnen una vez al día para cenar. El resto del tiempo lo pasan rezando a solas sin verse. No pueden hablar. El único que puede hacerlo es el abad.

Un día les dice: "Una terrible enfermedad no contagiosa ha llegado al monasterio y, desgraciadamente, veo que hay monjes infectados. El único síntoma que se puede apreciar es que al enfermo se le pone la cara negra. Sin embargo, él no sentirá nada. Quien contraiga la

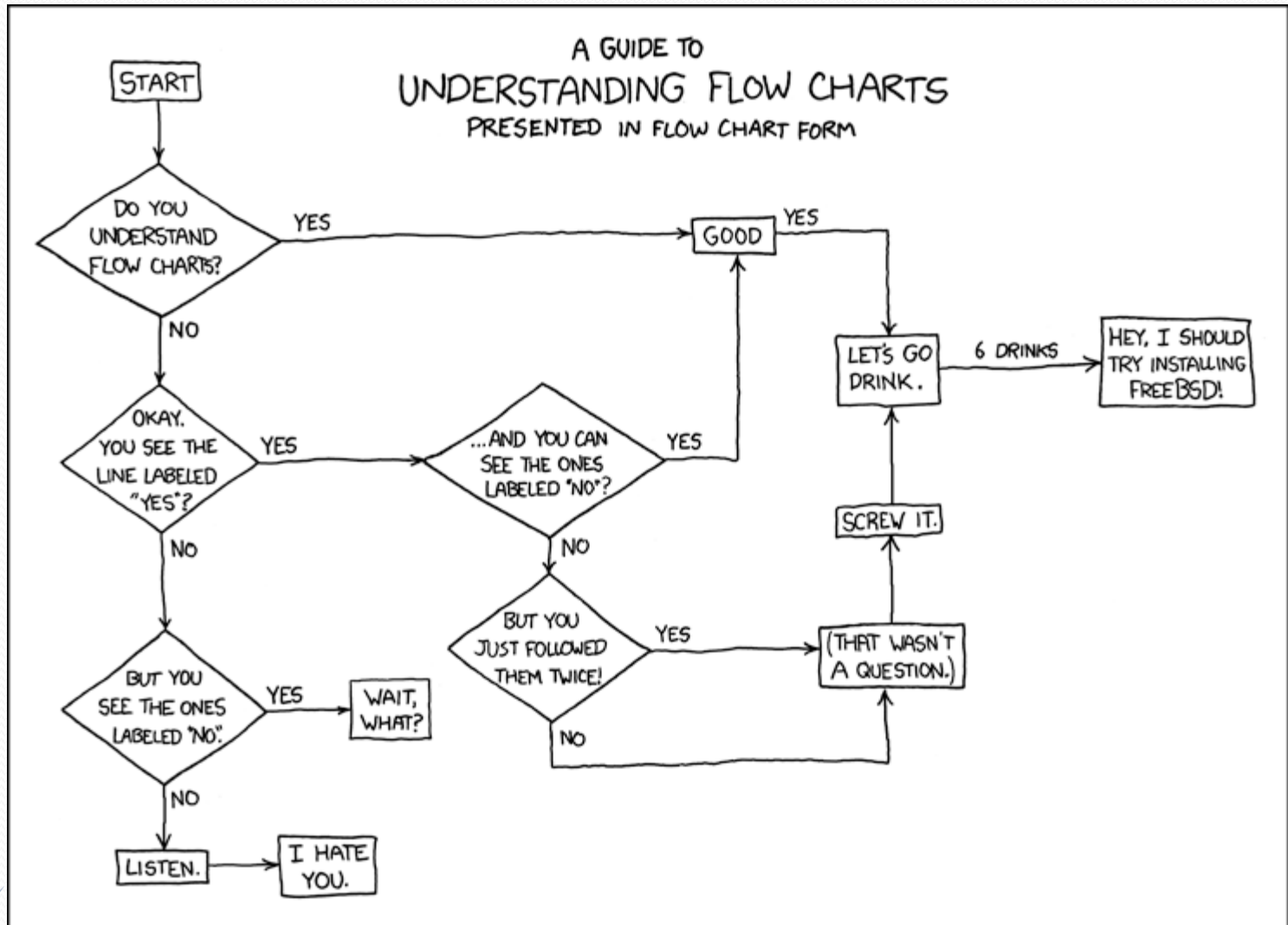
enfermedad debe suicidarse en cuanto lo sepa".

Los monjes siguen con su vida normal, hasta que N días después, al reunirse a cenar, ven que faltan algunos. Van a sus habitaciones y ven que se han suicidado, y que eran los que tenían la enfermedad. En el monasterio no hay espejos ni objetos reflectantes, por lo que los monjes no han podido verse la cara. Además, los monjes son unos racionalistas perfectos, y todos confían plenamente en la lógica de sus compañeros.

¿Cómo supieron los monjes infectados que efectivamente tenían la enfermedad?

¿Cuántos estaban enfermos?

Cuerpo del programa: Sentencias Condicionales



Cuerpo del programa: Sentencias Condicionales

.- Dependiendo del valor de una expresión lógica se ejecutan o no una o varias sentencias.

```
...  
if ( expresión lógica ) then  
  
    • sentencia 1  
    • ...  
    • sentencia n  
  
endif  
...
```

.- Si solo se va a ejecutar una sentencia puede usarse el siguiente formato

```
if ( expresión lógica ) sentencia 1
```

Cuerpo del programa:

Sentencias Condicionales (dobles)

```
...  
if ( expresión lógica ) then  
  
    • sentencia 1  
    • ...  
    • sentencia n  
  
else                ! Si no pasa (expresión lógica)  
  
    • sentencia 1  
    • ...  
    • sentencia n  
  
endif  
...
```

Cuerpo del programa:

Sentencias Condicionales (múltiples)- ¡¡ojo con el orden!!

```
if ( expresión lógica_1 ) then
```

- sentencia 1
- ...
- sentencia n

```
elseif ( expresión lógica_2 ) then
```

- sentencia 1
- ...
- sentencia n

```
•  
•  
•
```

```
elseif ( expresión lógica_m ) then
```

- sentencia 1
- ...
- sentencia n

```
endif
```

Ejercicio:

Escribir un programa que calcule y escriba el valor absoluto de un número real.

```
program absoluto

    implicit none

    ! Declaración de variables

    ! Inicialización de variables

    ! Cuerpo del programa

end program absoluto
```


Ejercicio:

Escribir un programa que calcule y escriba el valor absoluto de un número real.

```
program absoluto

    implicit none

    !Declaracion de variables
    real    :: x
    !Inicializacion de variables
    x=0.
    !Cuerpo de programa
    if (x>0) then
        X=+X
    else if (x<0) then
        X=-X
    end if
    write(*,*) x

end program absoluto
```

Ejercicio:

Escribir un programa que calcule y escriba el valor absoluto de un número real.

```
program absoluto

    implicit none

    !Declaracion de variables
    real    :: x

    !Inicializacion de variables
    x=0.

    !Cuerpo de programa
    if (x<0) x=-x

    write(*,*) x

end program absoluto
```

Ejercicio:

Escribir un programa que dado un número entero imprima “par” si el número es par o “impar” si es impar.

```
program paridad

    implicit none

    ! Declaración de variables

    ! Inicialización de variables

    ! Cuerpo del programa

end program paridad
```

Cuerpo del programa:

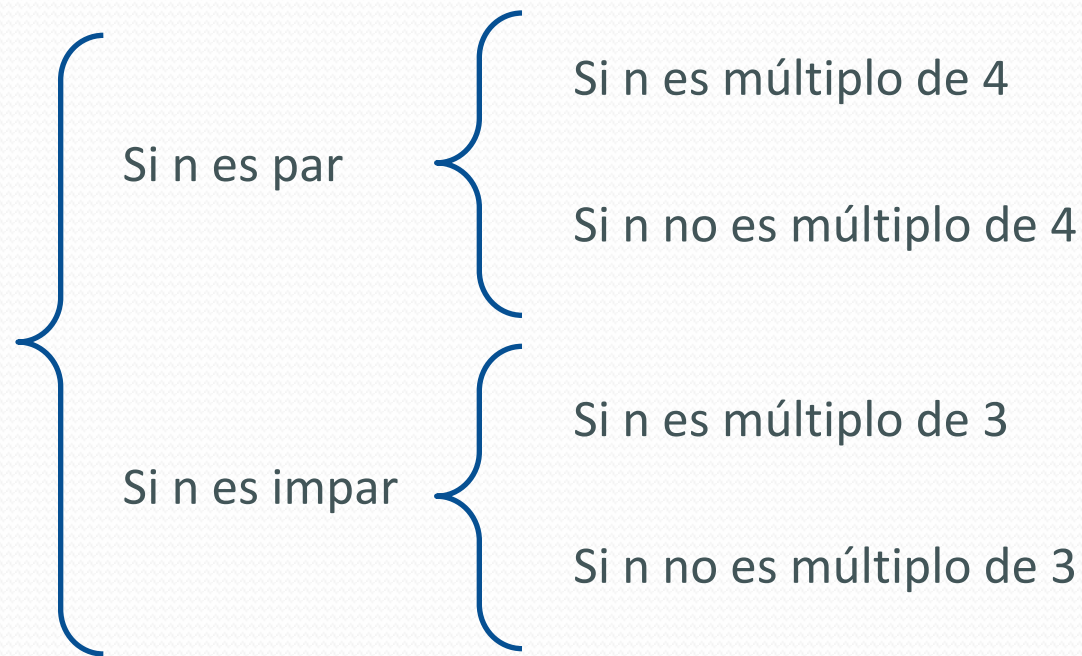
Sentencias Condicionales Anidadas

```
if ( expresión lógica_1 ) then  
    if ( expresión lógica_1_1 ) then  
        • sentencia 1  
        • ...  
        • sentencia n  
    elseif ( expresión lógica_1_2 ) then  
        • sentencia 1  
        • ...  
        • sentencia m  
    endif  
elseif ( expresión lógica_2 ) then  
    • sentencia 1  
    • ...  
    • sentencia n  
endif
```

Ejercicio:

Escribir un programa con las siguientes especificaciones:

Carga desde el teclado el valor de una variable n de tipo integer. Comprueba que es positivo e imprime, respecto de n ,



Problema para programar en casa

$$\text{Si } x < y \quad \begin{cases} f(x, y) = \text{sen}^2(x) & x < -2 \\ f(x, y) = \sqrt{x^2 + y^2} & -2 \leq x < 2 \\ f(x, y) = \frac{x}{2y} & 2 \leq x \leq 4 \\ f(x, y) = 7x^{4/3} & x > 4 \end{cases}$$

$$\text{Si } x > y \quad \text{e} \quad y > -7 \quad \begin{cases} f(x, y) = y - x & x < -2 \\ \begin{cases} f(x, y) = \frac{x}{y} & y \neq 0 \\ f(x, y) = 0 & y = 0 \end{cases} & -2 \leq x < 2 \\ f(x, y) = |y| & x \geq 2 \end{cases}$$

$$\text{En el resto de los casos} \quad f(x, y) = 7x^3 + 2x^2 - x + 5$$

Cuerpo del programa:

Sentencias Condicionales (múltiples `SELECT CASE`)

```
select case ( variable integer, character o logical)
            ! NO REAL
```

```
case (Rango_1 de valores para la variable)
```

- sentencia 1
- ...
- sentencia n

```
case (Rango_2 de valores para la variable)
```

- sentencia 1
- ...
- sentencia n

-
-

```
case default
```

- sentencia 1
- ...
- sentencia n

```
end select
```

Ejemplo: Sentencias Condicionales (múltiples `SELECT CASE`)

```
select case ( num )  
    case (:-1)  
        write(*,*) 'num <= -1'  
    case (0, 2, 4, 6, 8)  
        write(*,*) 'num es par y menor que 10'  
    case (1, 3, 5, 7, 9)  
        write(*,*) 'num es impar y menor que 10'  
    case (10:100)  
        write(*,*) '10 <= num <= 100'  
    case default  
        write(*,*) '100 < num'  
end select
```


Ejemplo: Sentencias Condicionales (múltiples `SELECT CASE`)

```
CHARACTER (LEN=1) :: c
```

```
SELECT CASE (c)
```

```
  CASE ('a' : 'j')
```

```
    WRITE(*,*) 'One of the first ten letters'
```

```
  CASE ('l' : 'p', 'u' : 'y')
```

```
    WRITE(*,*) 'One of l, m, n, o, p, u, v, w, x, y'
```

```
  CASE ('z', 'q' : 't')
```

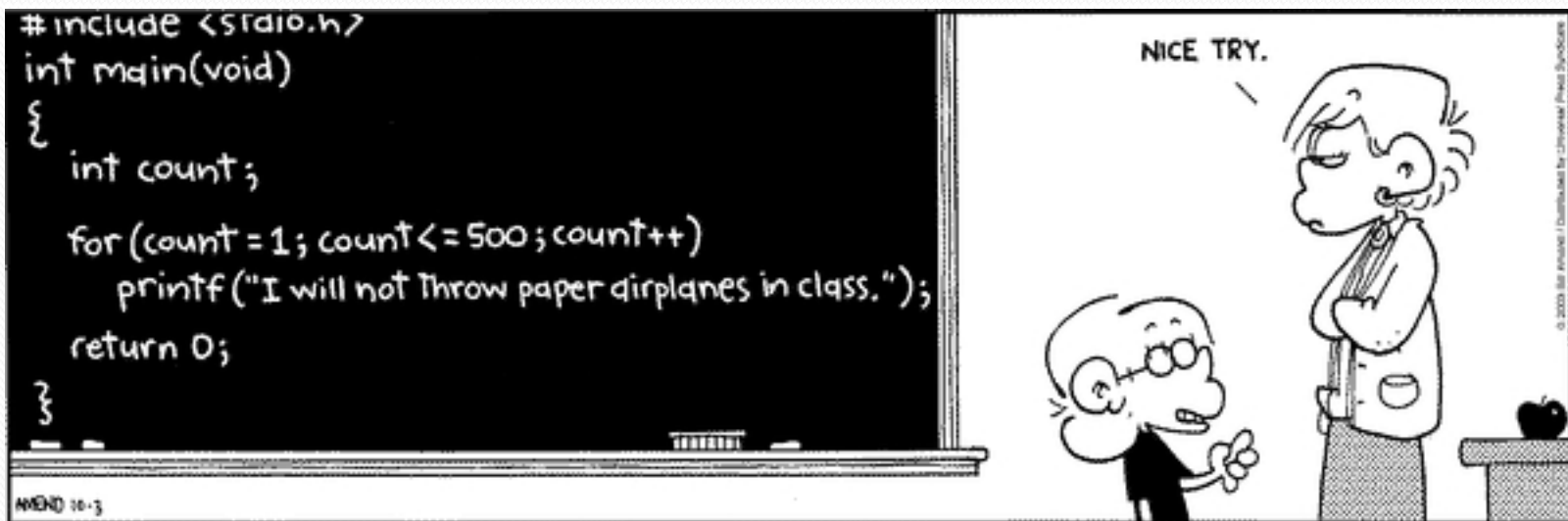
```
    WRITE(*,*) 'One of z, q, r, s, t'
```

```
  CASE DEFAULT
```

```
    WRITE(*,*) 'Other characters, which may not be letters'
```

```
END SELECT
```

Cuerpo del programa: Bucles iterados



Cuerpo del programa: Bucles iterados

Una estructura iterativa (*bucle*) es aquella que **ejecuta repetidas veces** un conjunto de sentencias (*rango del bucle*).

Existen dos tipos de estructuras iterativas:

- *Bucle controlado por un contador*

Utiliza un contador para fijar a priori el número máximo de iteraciones a realizar

- *Bucle controlado por una expresión lógica*

Una expresión lógica controla la salida del bucle

```
do XXXXX  
    sentencias  
enddo
```

Cuerpo del programa:

Bucle controlado por un contador

```
do i = Valor_inic, Valor_fin [, p]
```

```
    sentencia 1
```

```
    sentencia 2
```

```
    ...
```

```
    sentencia n
```

```
enddo
```

- i es una variable, llamada *contador*, de tipo *integer*.
- Valor_inic , Valor_fin , p son constantes o variables de tipo *integer*, tales que Valor_inic es el valor inicial del contador, Valor_fin es el valor final del contador y p es el incremento del contador en cada iteración.
- El incremento p no puede ser cero.
- Si $p = 1$, entonces no es necesario especificarlo.
- Si $p < 0$, entonces se debe cumplir $\text{Valor_inic} > \text{Valor_fin}$
- NUNCA modificar la variable contador dentro de un bucle.
- Al salir del bucle el valor del contador es el primero fuera del rango al incrementarle p

Ejemplos sencillos

```
program main
  integer :: i

  do i=1,3
    write(*,*) 'i vale:', i
  end do
  write(*,*) 'Fuera, i vale:', i

end program main
```

```
program main
  integer :: i

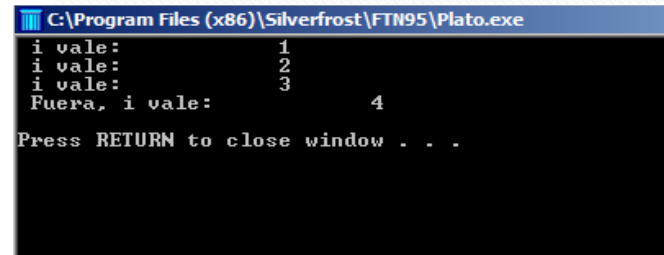
  do i=1,7,2
    write(*,*) 'i vale:', i
  end do
  write(*,*) 'Fuera, i vale:', i

end program main
```

Ejemplos sencillos

```
program main
  integer :: i

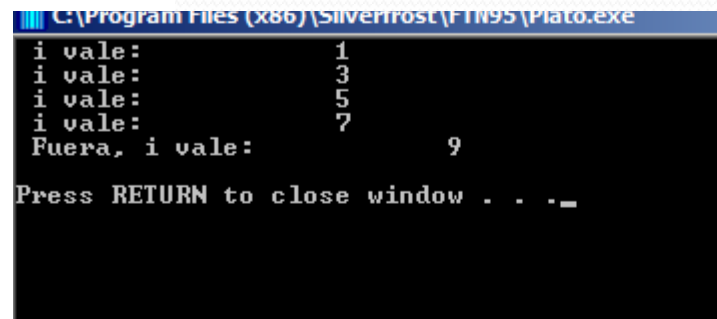
  do i=1,3
    write(*,*) 'i vale:', i
  end do
  write(*,*) 'Fuera, i vale:', i
end program main
```



```
C:\Program Files (x86)\Silverfrost\FTN95\Plato.exe
i vale: 1
i vale: 2
i vale: 3
Fuera, i vale: 4
Press RETURN to close window . . .
```

```
program main
  integer :: i

  do i=1,7,2
    write(*,*) 'i vale:', i
  end do
  write(*,*) 'Fuera, i vale:', i
end program main
```



```
C:\Program Files (x86)\Silverfrost\FTN95\Plato.exe
i vale: 1
i vale: 3
i vale: 5
i vale: 7
Fuera, i vale: 9
Press RETURN to close window . . .
```

Ejemplo: 2 bucles anidados

```
program main
  integer :: i, j

  do i=3,8,2
    write(*,*) 'i vale:', i

    do j=3,5
      write(*,*) 'j vale:', j
    end do
    write(*,*) 'Fuera, j vale:', j

  end do
  write(*,*) 'Fuera, i vale:', i
  write(*,*) 'Fuera, j vale:', j

end program main
```

Cc

Ejemplo: 2 bucles anidados

```
program main
  integer :: i, j

  do i=3,8,2
    write(*,*) 'i vale:', i

    do j=3,5
      write(*,*) 'j vale:', j
    end do
    write(*,*) 'Fuera, j vale:', j

  end do
  write(*,*) 'Fuera, i vale:', i
  write(*,*) 'Fuera, j vale:', j

end program main
```

C:\Program Files (x86)\Silverfrost\FTN95\Plato.exe

```
i vale: 3
  j vale: 3
  j vale: 4
  j vale: 5
    Fuera, j vale: 6
i vale: 5
  j vale: 3
  j vale: 4
  j vale: 5
    Fuera, j vale: 6
i vale: 7
  j vale: 3
  j vale: 4
  j vale: 5
    Fuera, j vale: 6
Fuera, i vale: 9
Fuera, j vale: 6

Press RETURN to close window . . .
```


Ejemplo: programa con sentencia if y bucle do

```
program sumatorio
  integer :: num
  integer :: suma
  integer :: i

  write(*,*) 'Introducir un numero > 0'
  read(*,*) num

  ! calculo de la suma 1 + 2 + ... + num

end program sumatorio
```

Ejemplo: programa con sentencia if y bucle do

```
program sumatorio
  integer :: num
  integer :: suma
  integer :: i

  write(*,*) 'Introducir un numero > 0'
  read(*,*) num

  ! calculo de la suma 1 + 2 + ... + num
  if (num > 0) then
    suma = 0
    do i=1,num
      suma = suma + i
    end do
    write(*,*) suma
  else
    write(*,*) 'numero incorrecto'
  end if

end program sumatorio
```

Ejemplo: programa con sentencia if y bucle do

```
program factorial
  integer :: num
  integer :: Fact
  integer :: i

  write(*,*) 'Introducir un numero >= 0'
  read(*,*) num

  ! calculo del factorial de num
  if (num >= 0) then
    Fact = 1
    do i=2,num
      Fact = Fact*i
    end do
    write(*,*) Fact
  else
    write(*,*) 'numero incorrecto'
  end if

end program factorial
```

Cuerpo del programa:

Bucle controlado por expresión lógica

Dos posibilidades:

OPCIONAL

```
do i = V_inic, V_fin [, p]
    sentencia 1
    ...
    if (condicion logica) exit
    ...
    sentencia n
enddo
```

Compilador standard:

```
do while (condicion logica)
    sentencia 1
    ...
    sentencia n
enddo
```

Ejercicio:

Escribir un programa con las siguientes especificaciones:

- Pide que se introduzca por teclado una variable n de tipo integer.
- Cuando el valor introducido es un numero primo imprime un mensaje por pantalla y para el programa.

```
write(*,*) 'Escribe un numero natural'  
read(*,*) n
```

Trabajo para casa

Escribe un programa que te calcule, con n un número entero que pidas por pantalla, los siguientes sumatorios

$$S_1 = \sum_{i=1}^n i^2$$

$$S_4 = \sum_{i=1}^n \sum_{j=i}^m (i * j + i - j)$$

$$S_2 = \sum_{i=1}^n (i + 1)^2$$

$$S_5 = \sum_{i=1}^n \sum_{j=i}^{n+m} (i - j)$$

$$S_3 = \sum_{i=1}^n \sum_{j=1}^m \frac{i + j}{i}$$

$$S_6 = \sum_{i=1}^n \sum_{j=0}^i \frac{(i - j)^2}{3}$$